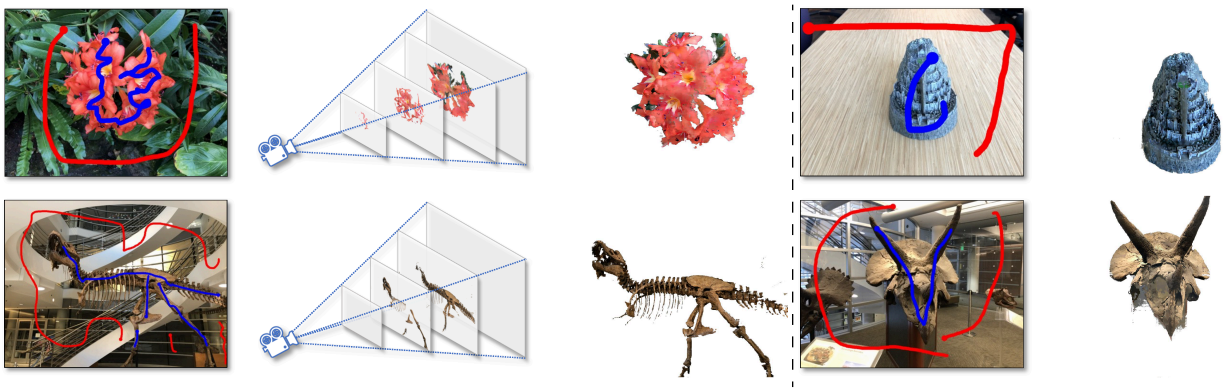
Neural Volumetric Object Selection

Zhongzheng Ren[1*] Aseem Agarwala[2†] Bryan Russell[2†] Alexander G. Schwing[1†] Oliver Wang[2†]
[1]University of Illinois at Urbana-Champaign [2]Adobe Research

https://jason718.github.io/nvos



2D image & scribble 3D object selection Novel view 2D image & scribble Novel view

Figure 1. We present an interactive 3D object selection method from neural volumetric representations (*e.g.*, MPIs [55] or NeRFs [33]). Blue/Red scribbles denote foreground/background. Please view with Adobe Acrobat or KDE Okular to see animations.

Abstract

We introduce an approach for selecting objects in neural volumetric 3D representations, such as multi-plane images (MPI) and neural radiance fields (NeRF). Our approach takes a set of foreground and background 2D user scribbles in one view and automatically estimates a 3D segmentation of the desired object, which can be rendered into novel views. To achieve this result, we propose a novel voxel feature embedding that incorporates the neural volumetric 3D representation and multi-view image features from all input views. To evaluate our approach, we introduce a new dataset of human-provided segmentation masks for depicted objects in real-world multi-view scene captures. We show that our approach out-performs strong baselines, including 2D segmentation and 3D segmentation approaches adapted to our task.

1. Introduction

Object selection is an important task in an artist's workflow. With the growth of 3D photography, there is a need to support a suite of editing operations, such as object selection, that are readily available for 2D photographs. In this work, we consider object selection in neural volumetric 3D representations. Neural volumetric 3D representations—explicitly via Multi-plane Images (MPI) [43] or implicitly via Neural Radiance Fields (NeRF) [33]—recover a remarkably accurate 3D representation of a scene from a given set of multi-view images. These representations have been shown to be particularly useful for novel-view synthesis [6,24,33,48,52,55], as this is the task that they are trained for. However, beyond just visualizing the same scene from a novel viewpoint, often we want to create a 3D reconstruction of an object so that we can extract it, and place it in a different context. There is, of course, substantial research in automatic object segmentation. However, user-driven methods, *e.g.*, Photoshop, remain the most common means, as *which* object to select is fundamentally a high-level decision. Similar user-driven methods for 3D object segmentation are particularly important for augmented and virtual reality (AR/VR) applications, *e.g.*, if one wants to composite a selected object into a new scene, apply a filter to a selected object, remove a selected object, or share a real-world object with friends in an AR/VR environment.

While user-driven segmentation for 2D images has been studied for decades [1, 11, 34, 40], very little is known about how segmentation would work in those novel 3D scene representations. Specifically, for image segmentation, early works focus on energy minimization with graph cuts [4], different forms of user interactions [4, 17, 35, 39], and ways

*Work partly done during an internship at Adobe Research.
†Alphabetic order.

to obtain better object priors [14, 45]. More recently, research has focused on encoding object priors with deep nets [23, 26–28, 49].

Naively applying these techniques to the set of images that are used to capture neural volumetric 3D representations is sub-optimal. For example, simply transferring user interactions like scribbles from one image to another using a known camera transformation may fail to cover the intended object because of occlusions. Similarly, transferring an appearance model learned on one image to all remaining images is challenging because of appearance and lighting changes. Asking a user to interact with all images requires an interface that may not be intuitive or require excessive work, and furthermore may produce a segmentation that is not view-consistent.

For those reasons, novel user-driven 3D segmentation techniques are warranted. We propose a novel voxel feature embedding that incorporates discretized features from the neural volumetric 3D representation and image features from all input views. Formally, we first project user interactions in the form of 2D scribbles from a reference image to sparse 3D locations using the reconstructed scene. We then learn a 3D object representation model that incorporates image features from all views via a developed multi-view feature embedding. We use these features to directly segment the object in the volumetric 3D scene representation and apply a post-processing step to remove outliers. The extracted 3D object can subsequently be viewed from different directions, as visualized in Fig. 1.

We evaluate the proposed method on real world samples from the LLFF [32], Shiny [48], and NeRF-real360 [33] data. As shown in Fig. 1, despite very few scribbles on a single reference image, the proposed method recovers an accurate 3D model of the object of interest and retains fine details (*e.g.*, the ribs of the dinosaur in row 1). To study quantitatively, we obtained annotations using a professional service. Our method out-performs 2D and 3D interactive segmentation baselines by a margin on all benchmarks.

In summary, we present the first method for user-driven 3D object selection targeting recent neural volumetric reconstruction. We show that a pre-trained network to embed multi-view features produces a more robust selection method than applying existing interactive 3D segmentation methods. For evaluation, we contribute a set of high-quality ground-truth annotations on three real-world datasets.

2. Related work

2D interactive segmentation approaches include binary object segmentation or alpha matting, which aim to estimate the proportion of two colors mixing to form a boundary. Early matting work goes back to the 1980s [1, 11] and the 1990s [34, 40]. Subsequent improvements like GrabCut [39] used a global energy minimization and popularized a simplification of the challenging matting task: first, estimate a "hard" segmentation; second, use border matting to compute alpha around a small strip at the boundary. A global energy minimization for interactive object segmentation has been used before by Boykov and Jolly [4] for object segmentation. Different forms of user input have been studied for interactive segmentation, including contours [17, 35], bounding boxes [39] and strokes [3, 4, 9, 13, 14, 21, 36, 37, 45, 46].

Beyond different user interactions, follow-up work has also focused on improving the energy function employed in the work by Boykov and Jolly [4] and GrabCut [39]. For example, Grady [13] studied random walks for speed-up and Veksler [45] and Gulshan *et al*. [14] introduced shape priors to incorporate more expressive object information.

More recently, more expressive object information has been incorporated via deep nets [23, 26–28, 49]. For instance, given user input, Xu *et al*. [49] finetune fully connected nets (FCNs), the output of which is then used in a graph-cut formulation. Subsequent work refined the predictions by incorporating diversity or attention [23, 26–28].

Different from the aforementioned works, we operate in a 3D volume rather than in image space. This setup requires developing a multi-view feature embedding which transforms scribbles from image space to volume space, and a pipeline that involves a 3D segmentation network, as well as a 3D graph-cut based post-processing step.

3D interactive segmentation is particularly common in the medical community. Indeed early image segmentation techniques were often developed with medical image segmentation in mind [4, 13]. Extending energy minimization techniques to 3D volumes proved to be challenging, necessitating various forms of improvements [13] to cope with increased memory requirements. Deep learning based techniques have been adopted more recently [25, 47], having the goal to capture object priors more accurately. Our focus is on visual realism instead of medical structure segmentation.

Image-based rendering (IBR) has a long history in computer graphics and computer vision [8], just like interactive segmentation. Image-based rendering aims to render novel views directly from input images and can be broadly categorized into methods which use geometry explicitly, methods which use geometry implicitly, and methods which do not use geometry at all. Classical techniques based on explicit geometry are texture-mapped models. Layered depth images (LDIs) [42], lumigraphs [5], flow-based [8], and tensor-based methods [2] implicitly use geometry, while light-field methods [20] try to avoid using geometry. Hybrid methods [10] have also long been studied.

More recently deep nets have been used for image-based rendering. Among the most widely used techniques are neural radiance fields (NeRF) [33] and multi-plane images (MPI) [55]. Both have in common that they aim to extract from a given set of images a volumetric representation of

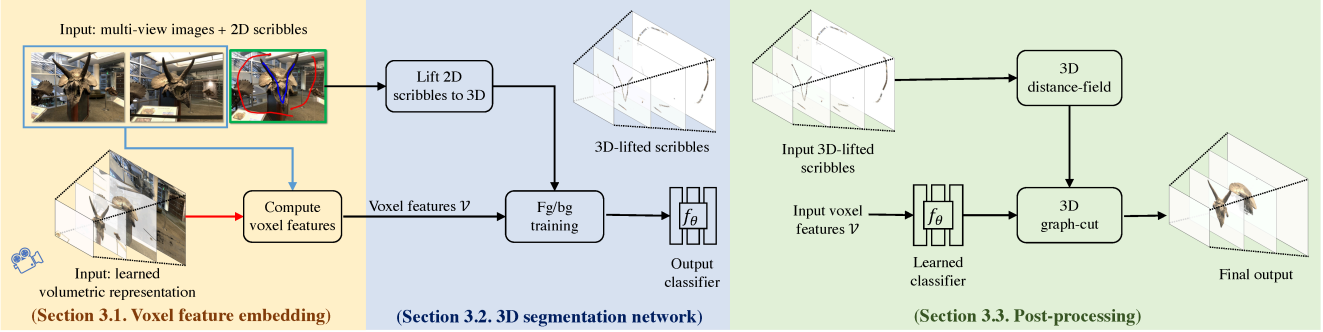


Figure 2. **Approach overview.** For each voxel in the 3D volume, we first compute a voxel feature embedding (**left**). We then train a 3D segmentation network to classify each voxel into foreground or background using as supervision the partial user scribbles (**center**). We apply the learned classifier and further refine the result using a 3D graph-cut which uses the 3D distance field of the scribbles (**right**).

the observed scene. While the volumetric representation is discrete for MPI and permits fast rendering, NeRF uses a continuous field to represent the volume.

Studying these representations for editing applications is an exciting direction. EditNeRF looked at modifying the colors and shape of a conditional radiance field given user scribbles [30]. Guo *et al*. [15] looked at composing photo-realistic scenes of captured objects. Yang *et al*. [50] leverage 2D object instance masks to train a compositional model for repositioning objects in a scene. Our goal is to use this volumetric representation for interactive object segmentation given user scribbles. We think these representations are particularly useful for this application and permit more accurate recovery of the object than the classical 2D and 3D interactive segmentation techniques discussed before. We note a contemporary effort that obtains (non-interactive) object segmentation in synthetic scenes by using unsupervised learning for object radiance fields [53]. Our work is the first that studies the use of these representations for interactive object selection in real-world scenes.

3. Approach

Given a set of multi-view images of a scene, the corresponding 3D volumetric representation, and a pair of 2D scribbles indicating the foreground and background in one specific view (which we refer to as the reference view), our goal is to segment the foreground object in the 3D volumetric representation. In this paper, we use 3D volumetric representations discretized from neural IBR models, *i.e*., MPI and NeRF, due to their high quality view-synthesis results.

This problem is challenging as user-provided 2D scribbles need to first be translated to sparse and possibly inaccurate 3D annotations. Equally important, the neural 3D volume representation of IBR models is often noisy as its geometry is never explicitly supervised during training. Instead, the volume is learned in an implicit manner through differentiable volume rendering. We find a classical 3D graph-cut solution struggles with these neural 3D volumetric representations, often failing or requiring frequent scene-specific parameter tuning. To address this failure, we propose a robust and expressive voxel feature embedding that incorporates multi-view features from the neural volumetric 3D representation and from image features from all input views. We find this feature representation allows us to train a simple yet accurate classifier, which can then be refined with a small amount of post-processing via graph-cuts.

Overview. Our method operates on a discretized 3D volume $\mathcal{V} = \{\mathbf{v}_p\}$ where $\mathbf{v}_p \in \mathbb{R}^C$ is the C-dimensional voxel feature embedding representing color and transparency of each 3D voxel location $p \in \mathbb{R}^3$. Given a set of 2D foreground and background scribbles in one specific view, we train a 3D segmentation network

$$f_\theta(\mathbf{v}_p) : \mathbb{R}^C \rightarrow [0, 1], \tag{1}$$

which predicts the foreground probability of each voxel p and which has learnable parameters θ. We first illustrate the voxel features, which are composed of a new feature derived from the multi-view input training images, its appearance as predicted by the IBR model (*e.g*., color, density, and maybe view-directional components), and the location of the voxel (§ 3.1). We then introduce the 3D segmentation network f_θ (§ 3.2). Lastly, we introduce the post-processing refinement (§ 3.3). We illustrate the overall method in Fig. 2.

3.1. Voxel feature embedding

To conduct 3D segmentation with only sparse and noisy 3D supervision, we need an expressive voxel representation that captures both 3D location and appearance information. We develop a novel representation $\mathbf{v}_p$ for each voxel p in the volume, which is the concatenation of three features,

$$\mathbf{v}_p = [\mathbf{v}_p^{\text{MVS}}; \mathbf{v}_p^{\text{IBR}}; \mathbf{v}_p^{\text{XYZ}}], \tag{2}$$

where $\mathbf{v}_p^{\text{MVS}}$ is our novel multi-view image feature embedding, $\mathbf{v}_p^{\text{IBR}}$ are discretized features extracted from IBR models, and $\mathbf{v}_p^{\text{XYZ}}$ is a 3D positional encoding. We illustrate the process to obtain the features in the yellow colored region of Fig. 2 and discuss it next.

Multi-view image features $\mathbf{v}_p^{\text{MVS}}$. The multi-view image features $\mathbf{v}_p^{\text{MVS}}$ encode appearance information from the observed views. We find this information to be particularly useful for user-driven segmentation. This is intuitive when considering the three limitations of features that are extracted from only an IBR volume: 1) features extracted from an IBR volume are particularly noisy since the geometry is learned implicitly through differentiable volume rendering and never explicitly supervised. They further degrade during discretization; 2) IBR volume voxels encode limited neighborhood information; and 3) IBR volume representations are learned to model appearance, which might be sub-optimal for recognition tasks. To address these three limitations, we develop the multi-view image feature $\mathbf{v}_p^{\text{MVS}}$ inspired by recent deep multi-view stereo (MVS) methods [6, 51]. Specifically, we encode multi-view images into the reference view using the following three steps:

• 2D feature extraction: for a multi-view image set $\{\mathbf{I}_i\}_{i=1}^M$ where M denotes the number of available views, a pre-trained 2D convolutional neural network (CNN) is used to extract image features $\{\mathbf{G}_i\}_{i=1}^M$ for each image i.

• Cost-volume construction: using the known camera intrinsic and extrinsic parameters $\{\mathbf{K}_i, \mathbf{R}_i, \mathbf{t}_i\}_{i=1}^M$, the extracted 2D feature maps $\{\mathbf{G}_i\}_{i=1}^M$ can be warped onto multiple 3D planes oriented fronto-parallel to the reference view $r \in \{1, \dots, M\}$, and their agreement can be recorded in a plane-sweep cost volume. Let (u, v) be a pixel location in the reference view r and $\mathbf{H}_{i\to r}(z)$ be a 3×3 homography matrix that projects a 2D homogeneous point in view i to the plane at depth z of reference view r. We obtain the warped feature map $\mathbf{G}_{i\to r,z}$ by applying the homography matrix to the pixel locations of the feature map $\mathbf{G}_i$ for view i,

$$\mathbf{G}_{i\to r,z}(u, v) = \mathbf{G}_i \left(\mathbf{H}_{i\to r}^{-1}(z)[u, v, 1]^T\right). \tag{3}$$

If $\mathbf{n}_r$ is the principal axis of the reference camera r, then the homography $\mathbf{H}_{i\to r}(z)$ is:

$$\mathbf{H}_{i\to r}(z) = \mathbf{K}_i\mathbf{R}_i \left(\mathbf{I} - \frac{(\mathbf{t}_r - \mathbf{t}_i)\mathbf{n}_r^T}{z}\right) \mathbf{R}_r^T\mathbf{K}_r^T. \tag{4}$$

We then aggregate the projected features from the M plane-sweep volumes into a single cost-volume via

$$\mathbf{G}_{r,z}(u, v) = \text{Var}_{i\in\{1,\dots,M\}}(\mathbf{G}_{i\to r,z}(u, v)), \tag{5}$$

where $\text{Var}(\cdot)$ calculates the variance of the features over the M feature maps. We calculate the variance as it explicitly measures the feature difference from multiple views, which has been validated to out-perform baselines calculating the features' mean [51]. The computed visual feature variance serves as a good indicator for probable surface locations and hence greatly informs the 3D segmentation.

• Feature computation: the computed cost-volume is often noisy due to occlusions or non-Lambertian reflectance. Therefore, we further refine it using a 3D U-Net [38]. Concretely, we compute the final feature volume via

$$\mathbf{v}_{(\cdot)}^{\text{MVS}} = g_\omega(\mathbf{G}_r), \tag{6}$$

where we concatenate each $\mathbf{G}_{r,z}$ along the Z-axis to form the 3D tensor $\mathbf{G}_r$ and g_ω is a 3D U-Net with parameters ω. To extract robust and expressive multi-view features, we adopt the learned weights ω from Chen *et al*. [6], who originally train the network g_ω on all scenes of the DTU dataset [16] for fast generalization of NeRF models to unseen scenes. Appendix Fig. A1 shows the details for the computation of $\mathbf{v}_p^{\text{MVS}}$.

Neural voxel features $\mathbf{v}_p^{\text{IBR}}$. We also extract voxel features from the neural IBR volume. To study the robustness and generalizability of our method, we use two recent IBR models: MPI and NeRF. An MPI is naturally a discretized volume for 3D scenes and we use the MPI variant NeX [48] here. In contrast, a NeRF is an implicit continuous neural representation and cannot be directly used. We hence adopt the PlenOctrees [52] discretization which is an octree-based representation that supports real-time rendering without compromising photometric quality. PlenOctrees convert a NeRF model to a regular volume of size 512^3. For the obtained volumes, both NeX and PlenOctrees leverage spherical basis functions for modeling the color information via

$$c_p(\mathbf{d}) = \sum_{l\in\{1,\dots,N\}} k_p^l H^l(\mathbf{d}), \tag{7}$$

where $c_p(\mathbf{d}) \in \mathbb{R}^3$ is the color of voxel p from view direction $\mathbf{d}$, $k_p^l \in \mathbb{R}^3$ for voxel p are RGB coefficients, $H^l(\mathbf{d})$ is a view-dependent basis function, and $l \in \{1, \dots, N\}$ is the basis function index. In addition, each voxel also stores one transparency value which we refer to as ξ_p (*e.g*., alpha-transparency in an MPI and density in a NeRF). The neural IBR feature is constructed as,

$$\mathbf{v}_p^{\text{IBR}} = [\xi_p, k_p^1, \dots, k_p^N], \tag{8}$$

where $[\cdot]$ denotes concatenation. Note that this feature is independent of the viewing direction $\mathbf{d}$.

Positional voxel features $\mathbf{v}_p^{\text{XYZ}}$. In addition to the multi-view image features, we also extract a positional encoding of the voxel location. For each voxel, we project its (x, y) location to a 40-dimensional feature vector using a positional encoding [33], and similarly its z location to a 16-dimensional feature vector. In total, for each voxel we obtain a 56 dimensional feature vector $\mathbf{v}_p^{\text{XYZ}}$.

3.2. 3D segmentation network

Given the voxel representation $\mathbf{v}_p \in \mathbb{R}^C$ detailed in § 3.1, we use a 3-layer MLP as the segmentation network f_θ to predict the foreground probability.

As user input is provided in the form of 2D scribbles on the reference view, to obtain training labels, we first lift the 2D foreground and background scribbles to 3D using the known camera pose. For this lifting, we define a 3D ray for each pixel and compute the intersecting 3D surface-voxel as the first voxel on the ray with accumulated transmittance

lower than $\gamma = 0.01$. This step yields an "expected depth" for each ray, which permits to assign to the corresponding voxel either a foreground or a background label. Our network is then trained for binary classification using a standard binary cross-entropy loss. This process is illustrated in the blue colored region of Fig. 2.

3.3. Post-processing

Since the 3D segmentation network (§ 3.2) is trained with limited supervision (sparse scribbles), the final prediction on the entire volume is occasionally noisy. There are two main causes for the noise: 1) **floaters**: these errors are generally isolated and far from the foreground scribbles in 3D space; and 2) **incompleteness**: surface voxels are predicted incorrectly if their appearance differs significantly from the foreground scribbles, even though these voxels are very close to the foreground scribbles in 3D space. The reason for these errors is that the classifier processes each voxel independently, *i.e.*, neighborhood correlations are not considered. To address this issue, we apply a distance field-based 3D graph-cut for post-processing. Note, the neural volumetric representations are often of high resolution, which graph-cut fails to process due to memory and computational limitations. To ensure fast inference, we operate in a down-sampled and truncated volume space and upsample the segmentation afterwards. Truncation removes planes that are unlikely to contain the object of interest. We down-sample the volume by $4\times$ on the XY-plane and truncate to 20 planes. Note that our initial 3D segmentation is in the original high-resolution volume.

We design the following energy function for 3D segmentation. As before, let p be a voxel in the 3D volume and $N \subseteq |\mathcal{V}| \times |\mathcal{V}|$ be a neighborhood system on the volume (we adopt a 6-connected neighborhood). We seek to infer the foreground/background label $y_p \in \{0, 1\}$ of each voxel p. Let the unary term ϕ_p indicate how likely a voxel p belongs to foreground or background and the pairwise term $\psi_{p,q}$ capture the correlation between voxel p and its neighboring voxel q. We minimize the sum of unary and pairwise terms,

$$E = \sum_{p \in \mathcal{P}} \phi_p(y_p) + \alpha \sum_{p,q \in N} \psi_{p,q}(y_p, y_q), \tag{9}$$

where α is a scalar balancing weight. We depict this process in the green colored region of Fig. 2.

The unary term is based on our network prediction and input scribbles, *i.e.*,

$$\phi_p(y_p) = \omega_1 \phi_p^c(y_p) + \omega_2 \phi_p^d(y_p). \tag{10}$$

Here, the first term relies on the segmentation network output and is formulated as

$$\phi_p^c(y_p) = \begin{cases} f_\theta(\mathbf{v}_p) & \text{if } y_p = 1 \\ 1 - f_\theta(\mathbf{v}_p) & \text{if } y_p = 0 \end{cases}, \tag{11}$$

which is the probability that a voxel p belongs to category y_p. The second term ϕ_p^d is based on a distance-field of voxel p to the input scribbles. Formally,

$$\phi_p^d(y_p) = \begin{cases} \min_{q \in \mathcal{F}} \text{Dist}(p, q) & \text{if } y_p = 1 \\ \min_{q \in \mathcal{B}} \text{Dist}(p, q) & \text{if } y_p = 0 \end{cases}, \tag{12}$$

where $\mathcal{B}, \mathcal{F}$ are the set of 3D-projected background and foreground scribbles and $\text{Dist}(\cdot)$ is a function computing the 3D distance between two voxels.

The pairwise term $\psi_{p,q}$ models correlation. For instance, if the feature encoding of two voxels are similar and if both voxels are close, we expect them to be labeled similarly. This correlation is formulated via the binary term

$$\psi_{p,q}(y_p, y_q) = |y_p - y_q| \cdot \text{Dist}(p, q)^{-1} \cdot \exp^{-\frac{(\mathbf{v}_p - \mathbf{v}_q)^2}{\sigma}}, \tag{13}$$

where σ is a positive scalar. This term is useful as the depth planes in the 3D volume may be irregularly spaced, *e.g.*, inverse depth spacing is also used for real-world forward facing scenes [48].

4. Experiments

In this section, we introduce our experimental setup and show quantitative results, followed by ablation studies and an analysis. Lastly, we provide a qualitative comparison.

Experimental setup. There are three stages: **1) training:** The classifier f_θ operates on only the features of voxels belonging to 3D-lifted scribbles, and is trained for binary classification (fg/bg). We annotate only one pair of scribbles (fg/bg) in the reference view. **2) inference:** The trained f_θ classifies all voxels in the entire 3D volume. No scribbles are needed. **3) evaluation:** with all voxels classified, we render the foreground voxel locations to a novel view for 2D evaluation (segmentation, photo-realism) relative to GT.

Datasets. We use three classical multi-view scene datasets with multiple objects. We test the MPI models on seven scenes from LLFF [32], which are front-facing real-world scenes, with 20-62 images each. We also test on four scenes from Shiny [48], which is captured in a similar manner as LLFF but contains more challenging view-dependent effects such as metallic and transparent objects. We test NeRF-based models (PlenOctrees [52]) on two real-world 360° scenes from NeRF [33] (NeRF-real360). Unfortunately, PlenOctrees does not generalize well to the front-facing datasets LLFF and Shiny due to the large scene depth range. We thus leave the task of 3D segmentation using NeRF models on front-facing scenes to future work. For all datasets, the input images are resized to a 1008×756 pixels following the common practice in novel-view synthesis [33, 48]. The camera parameters are estimated via Structure-from-Motion (SfM) using the publicly available COLMAP library [41].

Dataset	LLFF		Shiny	
Metrics	Acc.↑	IoU↑	Acc.↑	IoU↑
Random	50.0	13.5	50.0	13.5
Scribbles (projected)	8.1	16.6	8.0	20.7
2D segmentation (using projected scribbles)				
Graph-cut [19]	88.6	59.0	86.1	48.0
GrabCut [39]	78.1	49.0	66.2	31.1
DeepLabV3 [7]	91.5	56.6	88.6	50.4
DEXTR [31]	89.7	34.5	59.6	40.4
FCA-Net [29]	88.3	62.7	87.9	58.5
3D segmentation				
Graph-cut (3D)	73.6	39.4	78.3	32.4
Ours	**92.0**	**70.1**	**90.7**	**69.3**

Table 1. **2D mask evaluation results.** Our method yields more accurate object selection results on both datasets.

Scribbles. For evaluation, we annotate a fixed set of foreground-background scribbles per image as shown in Appendix Fig. A2 left column. For each scene, we have one pair of foreground ($\mathcal{F}$) and background ($\mathcal{B}$) scribbles for the reference view. Note that a scribble ($\mathcal{F}$ or $\mathcal{B}$) may contain several strokes. We use the scribbles as the input to our method. For baselines, we project these scribbles to the testing view using the recovered camera (see details in § 3.2). To test the generalizability, we ensure that the scribbles have different length and shape, sometimes covering multiple objects, *e.g.*, 'tools' and 'pasta' in the Shiny dataset.

Evaluation annotations & criteria. Our goal is to accurately segment 3D objects in volumetric representations. However, there is no standard way to annotate ground-truth 3D masks for real-world scenes since different IBR models represent 3D scenes differently. We propose two ways.

First, we evaluate the projected 2D mask segmentation results. For evaluation, we obtain high-quality 2D annotation masks in unseen validation views for the aforementioned three datasets using a professional image segmentation service. For each scene, one unseen validation image is annotated for evaluation. Note that the validation image view differs from the MPI reference view. Next, we render 2D foreground masks from the 3D volumetric representation in the novel view and compare to the corresponding 2D ground-truth. We report pixel classification accuracy (Acc) and foreground intersection-over-union (IoU).

Second, as our method operates on volumetric 3D representations, the segmented foreground object could be rendered to novel views through volume rendering. We thus report the photo-realistic quality of the segmented object rendered in a novel view with a black background. Since the objects could be small and we do not want the background to dominate the evaluation. To overcome this issue, we crop the foreground region on the rendered and ground-truth images using a tight bounding box around the ground-truth mask. We report Structural Similarity Index Measure (SSIM), Peak Signal-to-Noise Ratio (PSNR), and Learned Perceptual Image Patch Similarity (LPIPS) [54]. We do not report these metrics for the 2D baselines as they do not render new views.

Dataset	LLFF			Shiny		
Metrics	SSIM↑	PSNR↑	LPIPS↓	SSIM↑	PSNR↑	LPIPS↓
Graph-cut (3D)	0.600	15.03	0.415	0.454	12.83	0.477
Ours	**0.767**	**18.40**	**0.213**	**0.612**	**15.73**	**0.319**

Table 2. **Novel-view object rendering results.** Our model renders more realistic foreground objects than 3D graph-cut.

Baselines. We compare to several baselines: 1) **Random**: we randomly assign pixels to the two classes with equal probability. 2) **Scribbles (proj.)**: we lift the input 2D scribbles from the reference view into 3D and find the intersecting voxels (see details in § 3.2). We then project these voxels into the test image using the camera matrices. Since the projected scribbles are used as the input of baseline methods, this experiment helps to understand how accurate the scribbles are after projection. We visualize these projected scribbles in Appendix Fig. A2 right column.

Since the evaluation is conducted in 2D, we consider 2D interactive segmentation baselines. Specifically, given the input scribbles lifted to 3D and projected into the view that we have supervision for, we evaluate: **1) Graph-cut [19]**: we use the 2D Graph-cut with the unary term proposed in LazySnapping [22] and with an exponential binary term. The exact formulation is provided in the Appendix. **2) GrabCut [39]**: an improved version of Graph-cut based on iterative energy minimization. **3) DeepLabV3 [7]**: we fine-tune a state-of-the-art semantic segmentation net (DeepLabV3) for binary classification using the projected scribbles. **4) DEXTR [31]**: a pre-trained image interactive segmentation model that takes the 4 extreme points of the projected scribbles as input. **5) FCA-Net [29]**: a pre-trained 2D interactive image segmentation model that takes user clicks as input.

In contrast to 2D baselines, we aim to achieve 3D segmentation in volumetric representations rather than simply segmenting 2D objects in novel views. As a baseline for 3D interactive segmentation, we consider **Graph-cut (3D)**. Concretely, we keep everything the same as in § 3.3 except changing the first unary term (Eq. (11)) to

$$\phi_p^c(y_p) = \begin{cases} \min_{q\in\mathcal{F}} \|\mathbf{v}_p^{\text{IBR}} - \mathbf{v}_q^{\text{IBR}}\|_2 & \text{if } y_p = 1 \\ \min_{q\in\mathcal{B}} \|\mathbf{v}_p^{\text{IBR}} - \mathbf{v}_q^{\text{IBR}}\|_2 & \text{if } y_p = 0 \end{cases}, \tag{14}$$

where $\mathbf{v}_\cdot^{\text{IBR}}$ is given in Eq. (2).

Implementation details. The input images for both MPI and NeRF are resized to 1008×756 resolution. Following [48, 52], the MPI volume is of dimension $1156 \times 1408 \times 192$ and the PlenOctree volume is of size 512^3. Our cost volume is of size $640 \times 960 \times D$ where $D \in \{192, 512\}$ for MPI and PlenOctree, following MVS-Net [51]. Our classifier is implemented as a 2-layer MLP with hidden dimension 128. We train our network using the Adam optimizer with an initial learning rate of 0.001. During training, we adopt cross-validation where 10% of the scribbles' voxels are used

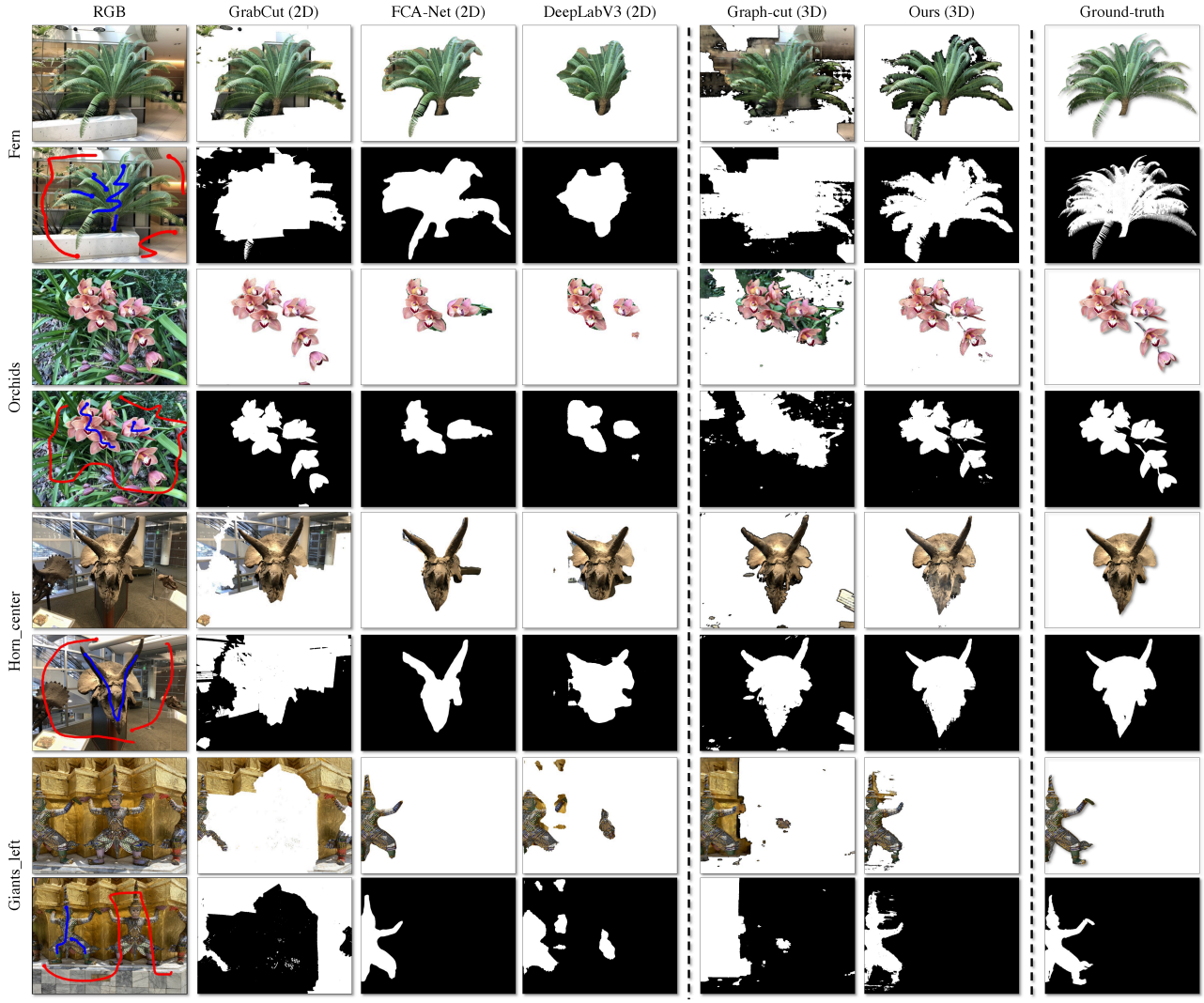


Figure 3. **Qualitative comparison.** For each scene, we show the reference view image and the input scribbles on the left. We then show the 2D mask and corresponding foreground image computed by different methods. On the right-most column, we show the ground-truth. Note that the foreground segmentation of the 2D baselines are not rendered; we apply the inferred mask to the ground truth test image.

as a validation set. Once the hyper-parameters are selected, we re-train our network with all scribbles' voxels to obtain the final model. For the 3D graph-cut parameters, we use $w_1 = 1, w_2 = 10, \alpha = 0.1, \sigma = 1$. Please refer to Appendix § A for more implementation details.

4.1. Quantitative results

2D mask evaluation. We report the 2D mask evaluation results in Tab. 1. We observe: 1) our model achieves higher segmentation accuracy than the best-performing optimization-based 2D interactive segmentation methods (Graph-cut or Grabcut) by 11.1%/21.3% IoU on the LLFF/Shiny datasets; 2) our model also improves upon the best-performing deep learning-based 2D interactive segmentation methods (DEXTR or FCA-Net) by 7.4%/10.8% IoU; 3) the 2D supervised segmentation model (DeepLabV3) makes accurate prediction (91.5%/88.6% Acc), but has significantly lower IoU (56.6%/50.4%) than ours (70.1%/69.3%). 4) our method also out-performs the 3D graph-cut baseline since it can only operate on a down-sampled volume and is not consistent across scene-specific neural volumetric representations.

Novel-view object rendering. Our model segments the foreground object in 3D and thus is able to render the object in novel views. We report the rendering results in Tab. 2 where we observe that our method significantly improves upon the 3D graph-cut baseline (+0.167 SSIM/+3.37 PSNR/-0.202 LPIPS on LLFF, and +0.158 SSIM/+2.90 PSNR/-0.158 LPIPS on Shiny).

4.2. Ablation study

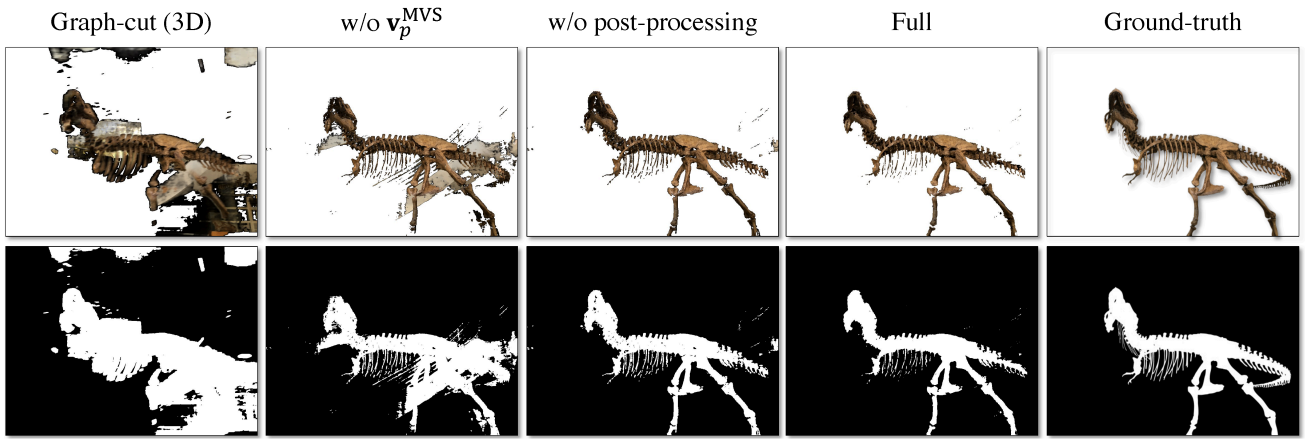


Figure 4. **Visual ablation.** Our method predicts more accurate and cleaner results than 3D segmentation baselines. Both the multi-view image feature $\mathbf{v}_p^{\text{MVS}}$ and the post-processing modules contribute to the final performance.

Metrics	Acc↑	IoU↑
w/o $\mathbf{v}_p^{\text{MVS}}$	87.6	64.7
w/o $\mathbf{v}_p^{\text{IBR}}$	89.1	60.4
w/o $\mathbf{v}_p^{\text{XYZ}}$	85.9	53.2
Kinetics 3D CNN	83.2	53.9
w/o post-proc.	90.9	68.0
Ours	**92.0**	**70.1**

Table 3. **Ablation study on LLFF.**

How important is each feature? One of our contributions is the voxel feature embedding detailed in § 3.1 which consists of three different terms (Eq. (2)). We validate the effectiveness of each one of them in Tab. 3 (top section). We observe that: 1) removing the introduced multi-view image feature embedding $\mathbf{v}_p^{\text{MVS}}$ hurts the performance by 5.4% IoU; 2) removing the learned feature $\mathbf{v}_p^{\text{IBR}}$ hurts the performance by 9.7% IoU; 3) removing the positional encoding $\mathbf{v}_p^{\text{XYZ}}$ yields the biggest IoU drop (16.9%).

How important is the 3D U-Net? In our experiments, we use the DTU [16] pre-trained 3D U-Net from Chen *et al.* [6] in Eq. (6). We verify the effectiveness of this pre-trained 3D U-Net by replacing it with a Kinetics [18] pre-trained 3D CNN encoder [44]. As shown in Tab. 3, replacing the U-Net hurts the results (-8.8% Acc, -16.2% IoU).

How important is the post-processing? We validate the effectiveness of post-processing in Tab. 3 where we find removing it degrades the results by 2.1% IoU.

4.3. Qualitative results

We present qualitative results in Fig. 3. Compared to both the 2D and 3D baselines, we observe our method to predict more accurate and complete foreground masks across objects. In addition, we also observe the proposed method to render foreground objects at testing views with good quality.

We further illustrate the effectiveness of the multi-view feature $\mathbf{v}_p^{\text{MVS}}$ and the post-processing in Fig. 4. Our method recovers fine-grained local details (*e.g.*, ribs) and significantly out-performs 3D graph-cut. The multi-view image feature $\mathbf{v}_p^{\text{MVS}}$ helps to better isolate the foreground object and post-processing helps visual smoothness, *i.e.*, it removes floaters.

5. Discussion

Limitations. We observe the proposed method to struggle in two main cases: 1) the final predictions still contain floaters (*e.g.*, Fig. 3 'Orchids' and 'Giants_left'). 2) our voxel feature may not capture view-dependent effects like reflections (*e.g.*, Fig. 3 'Horns_center' foreground image). Moreover, presently, our model does not run at interactive rates. While the features can be computed off-line, the segmentation network operates on the high-resolution volume which takes about 1-3min using our un-optimized implementation. The post-processing takes another 3-5min using un-optimized code. Faster feedback would make it easier for a user to determine where to add strokes. Finally, while our selections are currently binary, producing soft alpha mattes is an interesting area for future work.

As with most creative tools, object selection for compositing applications could be used for nefarious purposes, *e.g.*, to misinform. To combat this, we suspect that both forensics tools as well as a general increased cultural understanding of threats are needed to mitigate potential damages.

Conclusion. We present a user-driven way to segment objects in neural volumetric representations using user input on a single frame. For this, we develop robust features that can be classified accurately with a neural network. We further improve results with graph-cuts post-processing. We find the proposed method handles a variety of scenes well while scaling to high-resolution volumetric datasets. We believe that interactive segmentation in IBR volumes will be a key workflow in 3D asset generation and editing.

Acknowledgements: This work is supported in part by NSF #1718221, 2008387, 2045586, 2106825, MRI #1725729, NIFA 2020-67021-32799.

References

[1] John Adams, Milton Smith, and Paul Johnson. Spectral mixture modeling: A new analysis of rock and soil types at the Viking 1 lander site. *J. of Geophys. Res*, 1986. 1, 2

[2] Shai Avidan and Amnon Shashua. Novel view synthesis in tensor space. In *CVPR*, 1997. 2

[3] Xue Bai and Guillermo Sapiro. Geodesic matting: A framework for fast interactive image and video segmentation and matting. *IJCV*, 2009. 2

[4] Yuri Boykov and Marie-Pierre Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *ICCV*, 2001. 1, 2

[5] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured Lumigraph Rendering. *SIGGRAPH*, 2001. 2

[6] Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. MVSNeRF: Fast Generalizable Radiance Field Reconstruction from Multi-View Stereo. In *ICCV*, 2021. 1, 4, 8, 11

[7] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv:1706.05587*, 2017. 6, 12

[8] Shenchang E. Chen and Lance Williams. View Interpolation for Image Synthesis. *SIGGRAPH*, 1993. 2

[9] Antonio Criminisi, Toby Sharp, and Andrew Blake. Geos: Geodesic image segmentation. In *ECCV*, 2008. 2

[10] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and Rendering Architecture from Photographs: A hybrid geometry- and image-based approach. *SIGGRAPH*, 1996. 2

[11] Kenneth Fishkin and Brian Barsky. A family of new algorithms for soft filling. *Comp. Graph*, 1984. 1, 2

[12] Shanghua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip HS Torr. Res2net: A new multi-scale backbone architecture. *IEEE TPAMI*, 2019. 12

[13] Leo Grady. Random walks for image segmentation. *IEEE TPAMI*, 2006. 2

[14] Varun Gulshan, Carsten Rother, Antonio Criminisi, Andrew Blake, and Andrew Zisserman. Geodesic star convexity for interactive image segmentation. In *CVPR*, 2010. 2

[15] Michelle Guo, Alireza Fathi, Jiajun Wu, and Thomas Funkhouser. Object-centric neural scene rendering. *arXiv: 2012.08503*, 2020. 3

[16] Rasmus Jensen, Anders Dahl, George Vogiatzis, Engin Tola, and Henrik Aanæs. Large scale multi-view stereopsis evaluation. In *CVPR*, 2014. 4, 8

[17] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *IJCV*, 1988. 1, 2

[18] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv:1705.06950*, 2017. 8

[19] Vladimir Kolmogorov and Ramin Zabin. What energy functions can be minimized via graph cuts? *IEEE TPAMI*, 2004. 6

[20] Marc Levoy and Pat Hanrahan. Light field rendering. *SIGGRAPH*, 1996. 2

[21] Yin Li, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. Lazy snapping. *ACM TOG*, 2004. 2

[22] Yin Li, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. Lazy snapping. *ACM TOG*, 2004. 6, 11, 12

[23] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Interactive image segmentation with latent diversity. In *CVPR*, 2018. 2

[24] Zhengqi Li, Simon Niklaus, Noah Snavely, and Oliver Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. In *CVPR*, 2021. 1

[25] Xuan Liao, Wenhao Li, Qisen Xu, Xiangfeng Wang, Bo Jin, Xiaoyun Zhang, Yanfeng Wang, and Ya Zhang. Iteratively-Refined Interactive 3D Medical Image Segmentation with Multi-Agent Reinforcement Learning. In *CVPR*, 2020. 2

[26] Jun H. Liew, Scott Cohen, Brian Price, Long Mai, Sim-Heng Ong, and Jiashi Feng. MultiSeg: Semantically meaningful, scale-diverse segmentations from minimal user input. In *ICCV*, 2019. 2

[27] Jun H. Liew, Yunchao Wei, Wei Xiong, Sim-Heng Ong, and Jiashi Feng. Regional interactive image segmentation networks. In *ICCV*, 2017. 2

[28] Zheng Lin, Zhao Zhang, Lin-Zhuo Chen, Ming-Ming Cheng, and Shao-Ping Lu. Interactive image segmentation with first click attention. In *CVPR*, 2020. 2

[29] Zheng Lin, Zhao Zhang, Lin-Zhuo Chen, Ming-Ming Cheng, and Shao-Ping Lu. Interactive image segmentation with first click attention. In *CVPR*, 2020. 6, 12

[30] Steven Liu, Xiuming Zhang, Zhoutong Zhang, Richard Zhang, Jun-Yan Zhu, and Bryan Russell. Editing conditional radiance fields. In *ICCV*, 2021. 3

[31] Kevis-Kokitsi Maninis, Sergi Caelles, Jordi Pont-Tuset, and Luc Van Gool. Deep extreme cut: From extreme points to object segmentation. In *CVPR*, 2018. 6, 12

[32] Ben Mildenhall, Pratul P. Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local Light Field Fusion: Practical View Synthesis with Prescriptive Sampling Guidelines. *ACM TOG*, 2019. 2, 5, 13, 14

[33] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis. In *ECCV*, 2020. 1, 2, 4, 5, 13, 15

[34] Tomoo Mitsunaga, Taku Yokoyama, and Takashi Totsuka. AutoKey: Human assisted key extraction. *SIGGRAPH*, 1995. 1, 2

[35] Eric N. Mortensen and William A. Barrett. Intelligent scissors for image composition. *Proc. Computer graphics and interactive techniques*, 1995. 1, 2

[36] Brian L. Price, Bryan Morse, and Scott Cohen. Geodesic graph cut for interactive image segmentation. In *CVPR*, 2010. 2

[37] Zhongzheng Ren, Zhiding Yu, Xiaodong Yang, Ming-Yu Liu, Alexander G. Schwing, and Jan Kautz. UFO^2: A unified framework towards omni-supervised object detection. In *ECCV*, 2020. 2

[38] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015. 4

[39] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "GrabCut" – interactive foreground extraction using iterated graph cuts. *TOG*, 2004. 1, 2, 6, 12

[40] Mark A. Ruzon and Carlo Tomasi. Alpha Estimation in Natural Images. In *CVPR*, 2000. 1, 2

[41] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 5

[42] Jonathan Shade, Steven Gortler, Li-Wei Hey, and Richard Szeliski. Layered Depth Images. *SIGGRAPH*, 1998. 2

[43] Richard Szeliski and Polina Golland. Stereo matching with transparency and matting. In *ICCV*, 1998. 1

[44] Du Tran, Heng Wang, Lorenzo Torresani, Jamie Ray, Yann LeCun, and Manohar Paluri. A closer look at spatiotemporal convolutions for action recognition. In *CVPR*, 2018. 8

[45] Olga Veksler. Star shape prior for graph-cut image segmentation. In *ECCV*, 2008. 2

[46] Vladimir Vezhnevets and Vadim Konouchine. Growcut: Interactive multi-label nd image segmentation by cellular automata. In *Proc. Graphicon*, 2005. 2

[47] Guotai Wang, Maria A. Zuluaga, Wenqi Li, Rosalind Pratt, Premal A. Patel, Michael Aertsen, Tom Doel, Anna L. David, Jan Deprest, Sebastien Ourselin, and Tom Vercauteren. DeepIGeoS: a deep interactive geodesic framework for medical image segmentation. In *IEEE TPAMI*, 2018. 2

[48] Suttisak Wizadwongsa, Pakkapon Phongthawee, Jiraphon Yenphraphai, and Supasorn Suwajanakorn. NeX: Real-time View Synthesis with Neural Basis Expansion. In *CVPR*, 2021. 1, 2, 4, 5, 6, 11, 13

[49] Ning Xu, Brian Price, Scott Cohen, Jimei Yang, and Thomas Huang. Deep interactive object selection. In *CVPR*, 2016. 2

[50] Bangbang Yang, Yinda Zhang, Yinghao Xu, Yijin Li, Han Zhou, Hujun Bao, Guofeng Zhang, and Zhaopeng Cui. Learning object-compositional neural radiance field for editable scene rendering. In *ICCV*, 2021. 3

[51] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *ECCV*, 2018. 4, 6

[52] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. PlenOctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021. 1, 4, 5, 6, 13

[53] Hong-Xing Yu, Leonidas J. Guibas, and Jiajun Wu. Unsupervised discovery of object radiance fields. In *ICLR*, 2022. 3

[54] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 6

[55] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM TOG*, 2018. 1, 2

Appendix

In this appendix we provide:

§ A additional implementation details of our method.

§ B: additional implementation details of the baseline methods.

§ C: additional visualization of the input and lifted scribbles.

§ D: our NeRF-based model results.

§ E: discussion of the societal impact.

A. Implementation details

A.1. Multi-view image features $\mathbf{v}_p^{\text{MVS}}$

We illustrate the process of computing multi-view image features $\mathbf{v}_p^{\text{MVS}}$ in Fig. A1. We first extract 2D feature maps from multi-view images using a shared CNN model. Following [6], we use 3 close-by images as the multi-view set. Using more input images of different views is potentially beneficial, which we leave to future work. We then project the computed 2D feature map into the reference view using Eq. (3) and compute the variance cost volume. To align with the MPI model used in this paper, we use the same 192 depth planes as suggested in NeX [48]. Lastly, we extract the final features $\mathbf{v}_p^{\text{MVS}}$ using a 3D U-Net. All the network architectures are detailed in Tab. A1.

A.2. Post-processing

To ensure fast inference, post-processing operates in a down-sampled and truncated volume space. We down-sample the volume by $4\times$ (half the width and height) on the XY-plane using bi-linear interpolation. Based on the projected scribbles and the inferred 3D segmentation results, we first determine the null planes along the Z-axis where no foreground scribbles/predictions are located. Then for the remaining non-empty planes, we further down-sample to 20 planes using bi-linear interpolation over the Z-axis. After post-processing, we sample the 3D volume back to its original resolution. The 3D graph-cut baseline is computed in the same way.

B. Baselines

Graph-cut (2D). We use the unary term proposed in LazySnapping [22]. Let the sets $\mathcal{F}$ and $\mathcal{B}$ denote the projected foreground and background scribbles. We first run K-means in the color space of these two pixel sets to get K clusters. We refer to the cluster centers as $\mathbf{C}_k^{\mathcal{B}} \in \mathbb{R}^3$ and $\mathbf{C}_k^{\mathcal{F}} \in \mathbb{R}^3$ for cluster $k \in \{1, \ldots, K\}$. Then, for each pixel p, we compute the minimum distance from its color $\mathbf{c}_p \in \mathbb{R}^3$ to the foreground clusters as $d_p^{\mathcal{F}} = \min_k \|\mathbf{c}_p - \mathbf{C}_k^{\mathcal{F}}\|$; and similarly

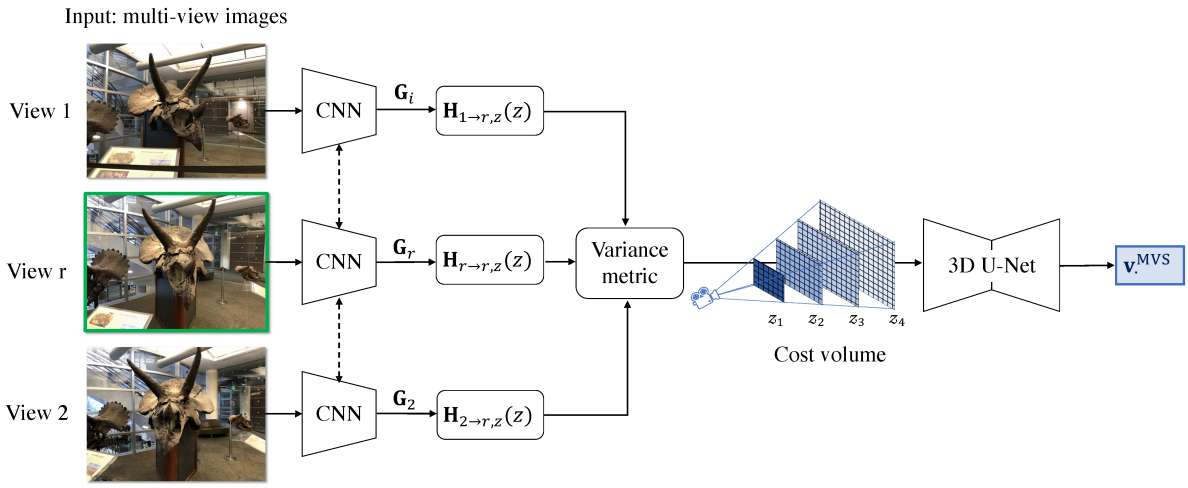


Figure A1. **Process of computing multi-view image features $\mathbf{v}_p^{\text{MVS}}$.**

#	Layer	Kernel	Stride	Dilation	Input channels	Output channels	Input
				Image encoder			
1	C2D/BN/ReLU	3×3	1	1	3	8	Image
2	C2D/BN/ReLU	3×3	1	1	8	8	#1
3	C2D/BN/ReLU	5×5	2	2	8	16	#2
4	C2D/BN/ReLU	3×3	1	1	16	16	#3
5	C2D/BN/ReLU	3×3	1	1	16	16	#4
6	C2D/BN/ReLU	5×5	2	2	16	32	#5
7	C2D/BN/ReLU	3×3	1	1	32	32	#6
8	C2D	3×3	1	1	32	32	#7
				3D U-Net			
9	C3D/BN/ReLU	3×3	1	1	(32+9)	8	#8+3×Images
10	C3D/BN/ReLU	3×3	2	1	8	16	#9
11	C3D/BN/ReLU	3×3	1	1	16	16	#10
12	C3D/BN/ReLU	3×3	2	1	16	32	#11
13	C3D/BN/ReLU	3×3	1	1	32	32	#12
14	C3D/BN/ReLU	3×3	2	1	32	64	#13
15	C3D/BN/ReLU	3×3	1	1	64	64	#14
16	CT3D/BN/ReLU	3×3	2	1	64	32	#15+#14
17	CT3D/BN/ReLU	3×3	2	1	32	16	#16+#12
18	CT3D/BN/ReLU	3×3	2	1	16	8	#17+#10

Table A1. Network architectures. C2D is a 2D convolutional layer, C3D is a 3D convolutional layer, CT3D is a 3D de-convolutional layer, BN is a batch-normalization layer.

$d_p^{\mathcal{B}} = \min_k \|\mathbf{c}_p - \mathbf{C}_k^{\mathcal{B}}\|$. The unary term is then defined as:

$$\phi_p(y_p) = \begin{cases} \phi_p(y_p = 0) = \infty & \forall p \in \mathcal{F} \\ \phi_p(y_p = 1) = 0 & \forall p \in \mathcal{F} \\ \phi_p(y_p = 0) = 0 & \forall p \in \mathcal{B} \\ \phi_p(y_p = 1) = \infty & \forall p \in \mathcal{B} \\ \frac{(1-y_p)d_p^{\mathcal{B}} + y_p d_p^{\mathcal{F}}}{d_p^{\mathcal{B}} + d_p^{\mathcal{F}}} & \text{otherwise} \end{cases} . \tag{A1}$$

For the binary term, we use the exponential term

$$\psi_{p,q}(y_p, y_q) = |y_p - y_q| \cdot \exp^{-\frac{(\mathbf{c}_p - \mathbf{c}_q)^2}{\sigma}}, \tag{A2}$$

where σ is a balancing term which we set to 10. Practically, we find this exponential term outperforms the one proposed in the LazySnapping [22] formulation.

GrabCut [39]. We use the GrabCut implementation provided in the OpenCV library.[1] We provide the projected foreground and background scribbles as input, and use the mask segmentation mode (`GC_INIT_WITH_MASK` in the OpenCV library) with 10 iterations.

DeepLabV3 [7]. We use the COCO pre-trained model with ResNet-50 backbone provided in the `torchvision` library.[2] We fine-tune the network for binary classification using a binary-cross-entropy loss, where the positive and negative examples are the projected foreground and background voxels respectively. We train the network for 20 epochs using Adam optimizer with a learning rate of 0.0001.

Deep Extreme Cut (DEXTR) [31]. We use the released model pre-trained on PASCAL VOC and SBD data. We compute the 4 extreme points from the projected foreground scribbles which are used as the network input.

FCA-Net [29]. We randomly sample 100 points from the projected foreground and background scribbles respectively, and process them with a pre-trained FCA-Net (Res2Net [12] backbone). We run the baseline model 5 times and report the mean value.

[1] https://docs.opencv.org/3.4/d8/d83/tutorial_py_grabcut.html
[2] https://pytorch.org/vision/stable/models.html

C. Additional visualization of the scribbles

For completeness, we illustrate the reference image and the input scribbles, together with the test-view image and the lifted scribbles in Fig. A2.

D. NeRF results

Our method also generalize to NeRF models. We use PlenOctrees [52] to extract the discrete volumetric representation from learned NeRF networks. For our purpose, we use the NeRF-real360 dataset which contains 2 real-world 360 scenes. Following PlenOctrees [52], we use the modified NeRF models where Spherical Harmonics are used to represent color rather than RGB values. We then convert the learned NeRF network into a 512^3 volume following the suggested PlenOctrees settings [52]. We present the qualitative results in Fig. A3 where our method correctly localizes and segments the foreground object in the scene.

E. Societal impact

In this work, we study novel-view object selection in neural volumetric representations. Our approach has the potential to positively impact applications in computer graphics and augmented reality, among them applying artistic effects on selected objects. However, our approach could also be used as a component for compositing objects into 3D scenes to create misinformation.

This research does not use human-derived data. The LLFF [32] dataset is released under the GNU General Public License v3.0. The Shiny [48] and NeRF-real 360 [33] datasets are released under MIT License. These datasets are mainly real-world scenes and do not contain offensive content or involve high-risk groups.

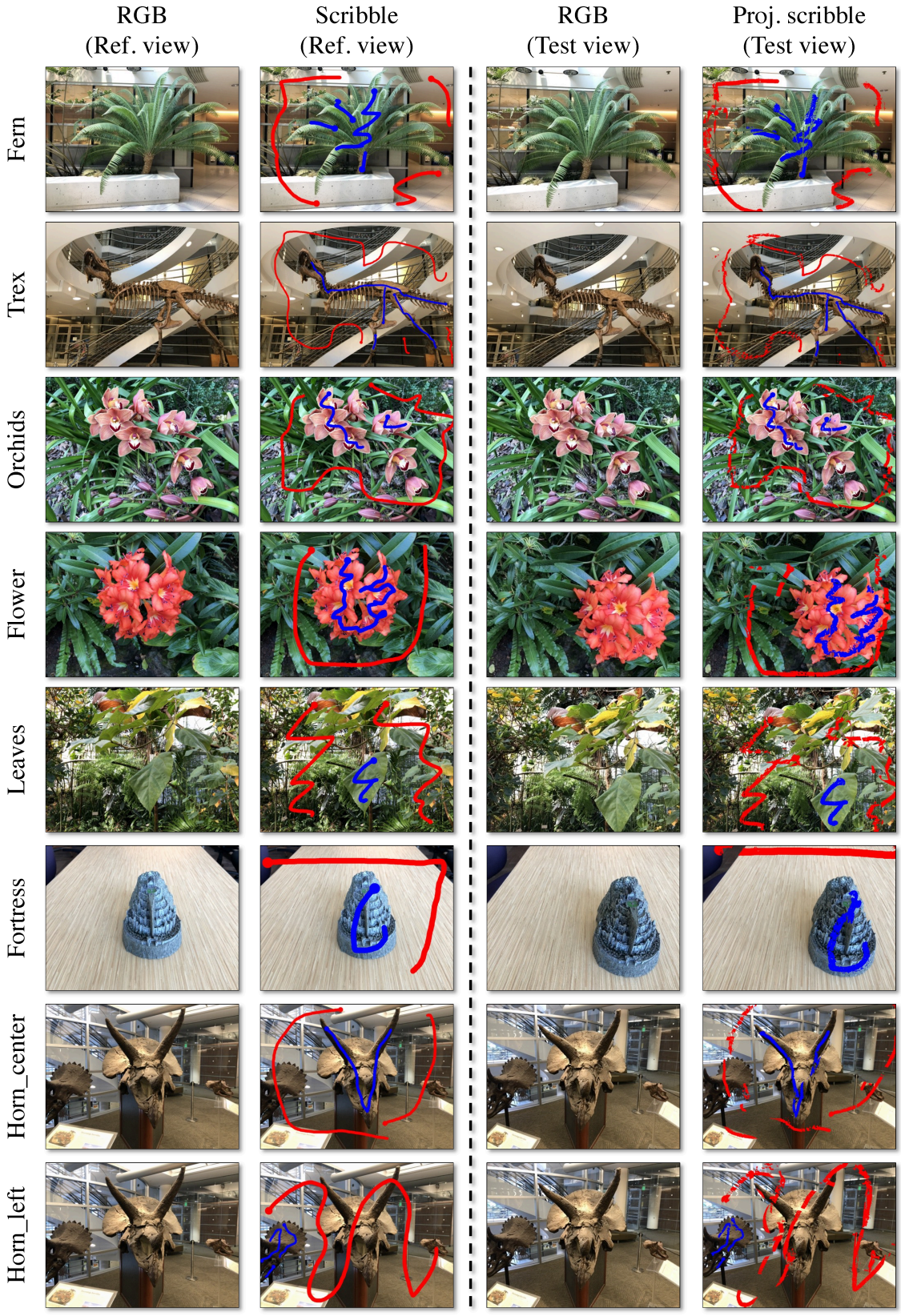


Figure A2. Examples of the original and projected scribbles on the LLFF [32] dataset (blue: foreground, red: background).

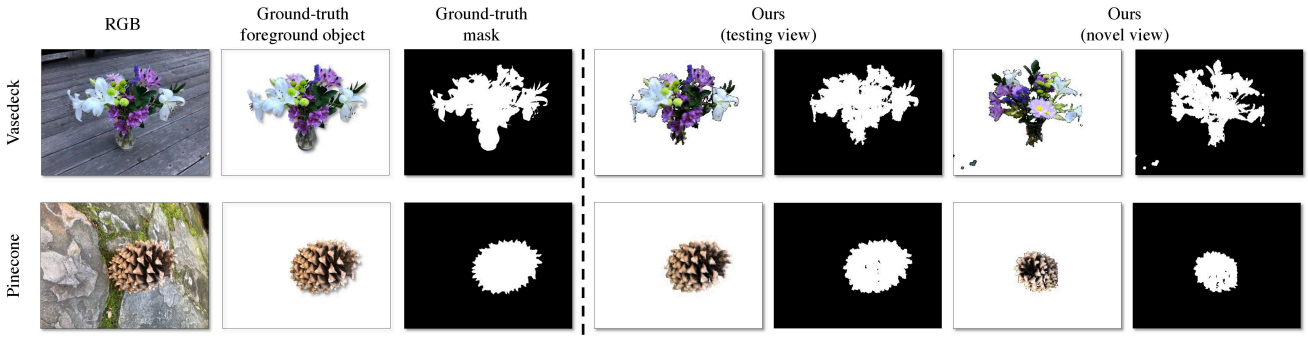


Figure A3. Qualitative results on the NeRF-real360 [33] dataset.